

Algorithmique et Programmation

Introduction aux classes

IUT Informatique de Bordeaux

Plan du cours

- 1 Types de base : les limites
- 2 Déclaration et initialisation
- 3 Références

Types primitifs : rappels

Jusqu'ici nous avons utilisé 4 types de base en Processing :

- `int`
- `float`
- `char`
- `boolean`

(et parfois `color`, qui encode un `int`)

Types primitifs : rappels

Jusqu'ici nous avons utilisé 4 types de base en Processing :

- `int`
- `float`
- `char`
- `boolean`

(et parfois `color`, qui encode un `int`)

Se limiter à ces types pose quelques problèmes.

Types primitifs : limites en Processing

Lisibilité

En Processing,

- beaucoup de variables accessibles dans tout le programme, et
- beaucoup de paramètres dans les fonctions.
- exemple : tester si 2 rectangles se touchent : 8 paramètres.

```
1 boolean intersectionRect(int r1x, r1y, r1w, r1h, r2x, r2y, r2w, r2h) { ... }
```

Types primitifs : limites en Processing

Entrée/sortie

Le passage de paramètre par valeur permet de passer des variables

- en **entrée** (lecture),
- mais **pas** en **entrée/sortie** (lecture/écriture).

Rappelez-vous... (cf slide suivant)

La transmission par valeur

```
1 void changer(int first, int b)
2 {
3     int temp;
4     temp=first;
5     first=b;
6     b=temp;
7     println("Dedans : first vaut ", first, " et b vaut ", b);
8 }
9
10 void setup()
11 {
12     int first=3;
13     int last=5;
14     println("Avant : first vaut ", first, " et last vaut ", last);
15     changer(first, last);
16     println("Après : first vaut ", first, " et last vaut ", last);
17 }
```

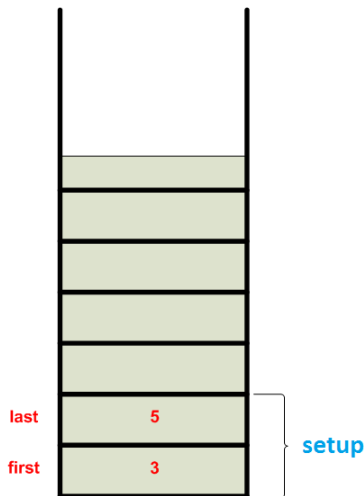
```
Avant : first vaut 3 et last vaut 5
Dedans : first vaut 5 et last vaut 3
Après : first vaut 3 et last vaut 5
```

La transmission par valeur

Exemple

```
void changer(int first, int b)
{
    int temp;
    temp=first;
    first=b;
    b=temp;
    println("Dedans : first vaut ", first, " et b vaut ", b);
}

void setup()
{
    int first=3;
    int last=5;
    println("Avant : first vaut ", first, " et last vaut ", last);
    changer(first, last);
    println("Après : first vaut ", first, " et last vaut ", last);
}
```

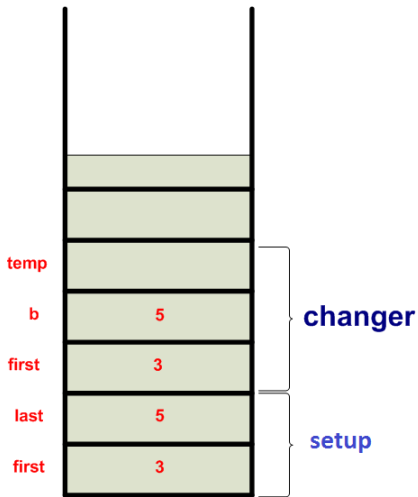


La transmission par valeur

Exemple

```
void changer(int first, int b)
{
    int temp;
    temp=first;
    first=b;
    b=temp;
    println("Dedans : first vaut ", first, " et b vaut ", b);
}

void setup()
{
    int first=3;
    int last=5;
    println("Avant : first vaut ", first, " et last vaut ", last);
    changer(first, last);
    println("Après : first vaut ", first, " et last vaut ", last);
}
```

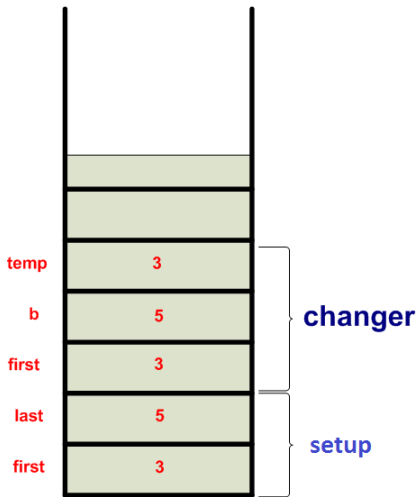


La transmission par valeur

Exemple

```
void changer(int first, int b)
{
    int temp;
    temp=first;
    first=b;
    b=temp;
    println("Dedans : first vaut ", first, " et b vaut ", b);
}

void setup()
{
    int first=3;
    int last=5;
    println("Avant : first vaut ", first, " et last vaut ", last);
    changer(first, last);
    println("Après : first vaut ", first, " et last vaut ", last);
}
```

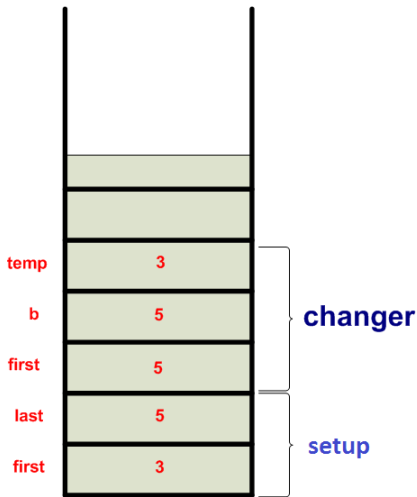


La transmission par valeur

Exemple

```
void changer(int first, int b)
{
    int temp;
    temp=first;
    first=b;
    b=temp;
    println("Dedans : first vaut ", first, " et b vaut ", b);
}

void setup()
{
    int first=3;
    int last=5;
    println("Avant : first vaut ", first, " et last vaut ", last);
    changer(first, last);
    println("Après : first vaut ", first, " et last vaut ", last);
}
```

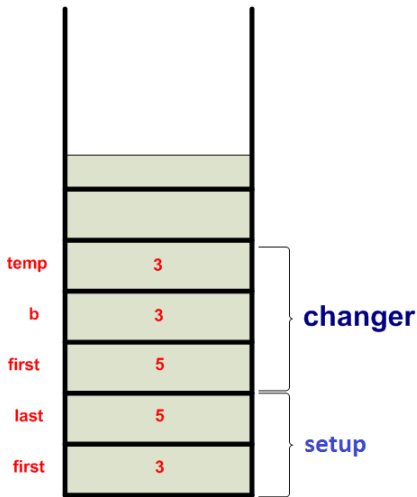


La transmission par valeur

Exemple

```
void changer(int first, int b)
{
    int temp;
    temp=first;
    first=b;
    b=temp;
    println("Dedans : first vaut ", first, " et b vaut ", b);
}

void setup()
{
    int first=3;
    int last=5;
    println("Avant : first vaut ", first, " et last vaut ", last);
    changer(first, last);
    println("Après : first vaut ", first, " et last vaut ", last);
}
```

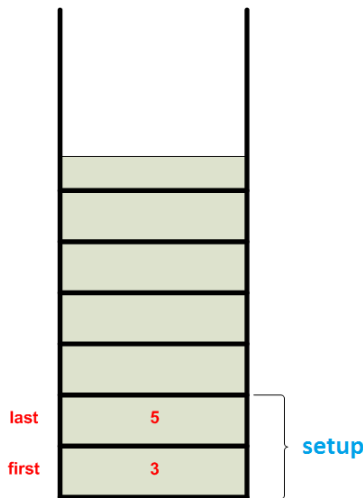


La transmission par valeur

Exemple

```
void changer(int first, int b)
{
    int temp;
    temp=first;
    first=b;
    b=temp;
    println("Dedans : first vaut ", first, " et b vaut ", b);
}

void setup()
{
    int first=3;
    int last=5;
    println("Avant : first vaut ", first, " et last vaut ", last);
    changer(first, last);
    println("Après : first vaut ", first, " et last vaut ", last);
}
```



Classes : intérêt

Les classes vont nous permettre (notamment) de :

- **regrouper** les données, et
- les passer aux fonctions en **entrée/sortie**.

Plan du cours

- 1 Types de base : les limites
- 2 Déclaration et initialisation
- 3 Références

Classe : définition

Une classe permet de regrouper des informations concernant une “entité” (exemple ici : une salle, ou un rectangle).

Définir une classe

```
1  class Salle {  
2      int capacite;  
3      boolean salleMachine;  
4      boolean libreService;  
5      int numero;  
6  }
```

```
1  class Rectangle {  
2      int x;  
3      int y;  
4      int largeur;  
5      int hauteur;  
6  }
```

Comme pour les fonctions, ces déclarations se font n'importe où dans le code, sauf à l'intérieur de fonctions / classes / etc.

Vocabulaire

Les champs `capacite`, `salleMachine`, `salleLibreService` et `numero` sont les **attributs** de la **classe** `Salle`.

Classe : définition

Une classe n'est pas un fourre-tout !

On y regroupe les variables qui concernent la même chose, pas “toutes les variables du programme”.

Classe : définition

Définir une classe

```
1  class Salle {  
2      int capacite;  
3      boolean salleMachine;  
4      boolean libreService;  
5      int numero;  
6  }
```

```
1  class Rectangle {  
2      int x;  
3      int y;  
4      int largeur;  
5      int hauteur;  
6  }
```

Une classe constitue un **nouveau type** de données.
On peut donc déclarer des variables de ce type :

Déclarer une variable

```
1  Salle salle1;  
2  Salle salle2;
```

```
1  Rectangle r1;  
2  Rectangle r2;
```

Classe : définition

Déclarer une variable

```
1 Salle salle1;
```

Vocabulaire

- avant son initialisation, cette variable prend la valeur **null**.
- une fois initialisée, la variable `salle1` contient une **référence** vers une **instance** de la **classe** `Salle` (nous verrons pourquoi ensuite).

On dit aussi que `salle1` est un **objet** de **type** `Salle`.

Classe : initialisation

Pour initialiser une instance, on utilise un **constructeur**.
Par défaut, il y a toujours un constructeur sans paramètre :

Constructeur par défaut

```
1 Salle salle1;  
2 Salle salle2;  
3 salle1 = new Salle();  
4 salle2 = new Salle();
```

ou

```
1 Salle salle1 = new Salle();  
2 Salle salle2 = new Salle();
```

Ensuite on accède à chaque attribut avec la notation
`instance.attribut` :

Accès aux attributs

```
1 salle1.capacite = 30;  
2 salle1.salleMachine = true;  
3 salle1.libreService = true;  
4 salle1.numero = 301;
```

```
1 salle2.capacite = 30;  
2 salle2.salleMachine = true;  
3 salle2.libreService = false;  
4 salle2.numero = 303;
```

Initialisation du contenu de l'instance

On peut également définir son propre constructeur. Nous le verrons lors de la prochaine séance. Aujourd'hui nous allons nous contenter d'initialiser chaque attribut avec une fonction.

Créer une fonction initSalle

```
1  Salle initSalle(int uneCapacite, boolean estUneSalleMachine,  
2                    boolean estEnLibreService, int unNumero) {  
3      Salle uneSalle = new Salle();  
4      uneSalle.capacite = uneCapacite;  
5      uneSalle.salleMachine = estUneSalleMachine;  
6      uneSalle.libreService = estEnLibreService;  
7      uneSalle.numero = unNumero;  
8      return uneSalle;  
9  }
```

```
1  Salle salle1 = initSalle(30, true, true, 301);
```

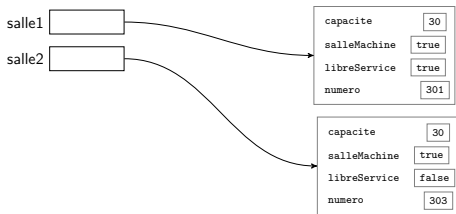
Plan du cours

- 1 Types de base : les limites
- 2 Déclaration et initialisation
- 3 **Références**

Références

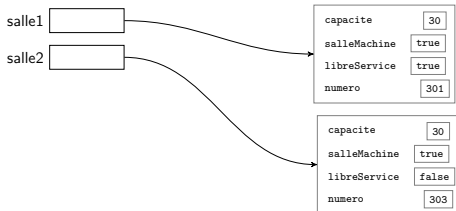
En Java (et donc Processing), on manipule toujours des **références** vers les objets (instances) :

```
1 Salle salle1 = initSalle( 30, true, true, 301);  
2 Salle salle2 = initSalle( 30, true, false, 303);
```

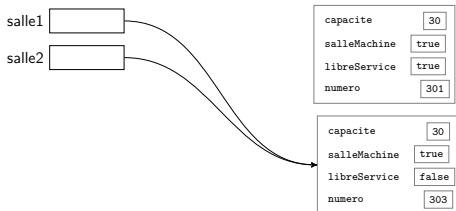


Références

```
1 Salle salle1 = initSalle( 30, true, true, 301);  
2 Salle salle2 = initSalle( 30, true, false, 303);
```




```
1 salle1 = salle2;
```



Passage de référence en paramètre

Une conséquence est que les **objets passés en paramètre** d'une fonction peuvent être **modifiés** (lecture / écriture).

```
1  /**
2   * Demenage une salle d'un etage vers un nouvel etage.
3   */
4  void demenage(Salle s, int etageAvant, int etageApres) {
5      // si la salle est a l'etage concerne
6      if (s.numero / 100 == etageAvant) {
7          // on demenage ! 301 devient 201
8          s.numero = (s.numero % 100) + (etageApres * 100);
9      }
10 }
11 void setup() {
12     Salle salle1 = initSalle( 30, true, true, 301);
13     Salle salle2 = initSalle( 30, true, false, 303);
14     int etageSrc = 3;
15     int etageDst = 2;
16     println("Numero avant :", salle1.numero);
17     demenage(salle1, etageSrc, etageDst);
18     println("Numero apres :", salle1.numero);
19 }
```



```
Numero avant : 301
Numero apres : 201
```

Console

Erreurs

→ pourquoi ?

Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}
```

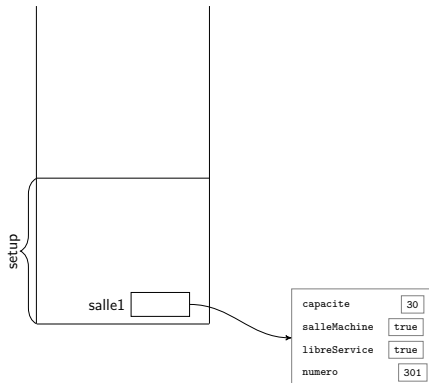
```
void setup() {  
    Salle salle1 = initSalle( 30, true, true, 301);  
    Salle salle2 = initSalle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```

setup



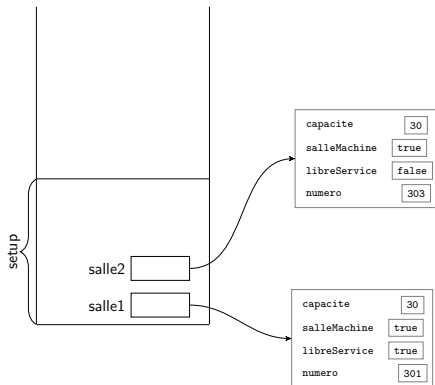
Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = initSalle( 30, true, true, 301);  
    Salle salle2 = initSalle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



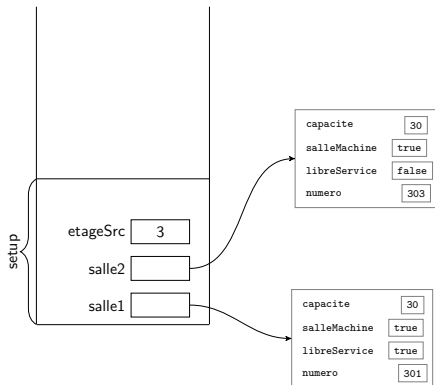
Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = initSalle( 30, true, true, 301);  
    Salle salle2 = initSalle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



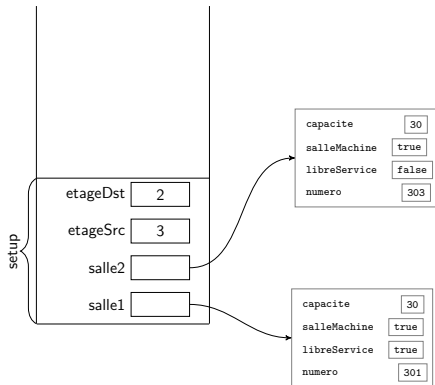
Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = initSalle( 30, true, true, 301);  
    Salle salle2 = initSalle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



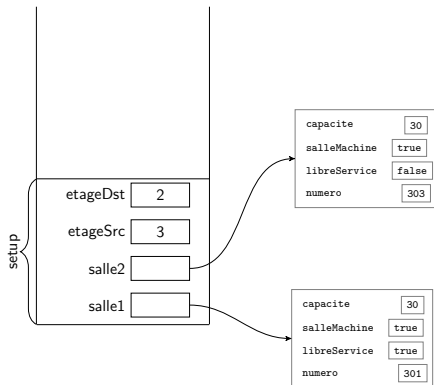
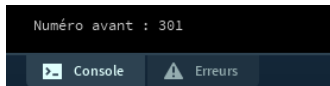
Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = initSalle( 30, true, true, 301);  
    Salle salle2 = initSalle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



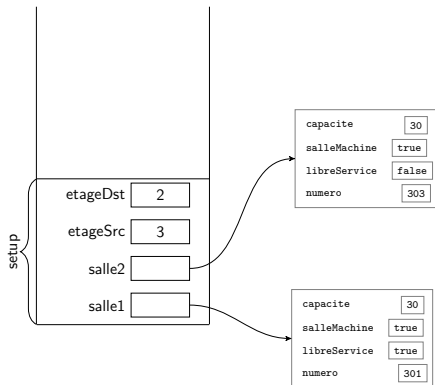
Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = initSalle( 30, true, true, 301);  
    Salle salle2 = initSalle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



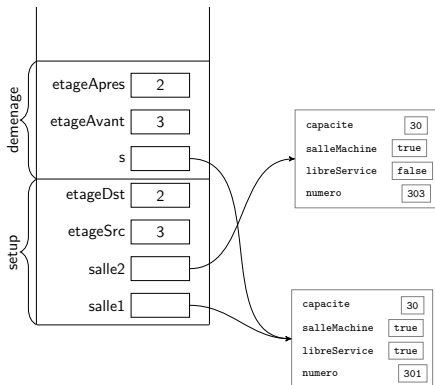
Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = initSalle( 30, true, true, 301);  
    Salle salle2 = initSalle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



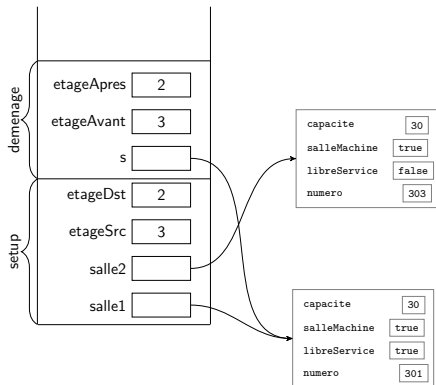
Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageAprès) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageAprès * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = initSalle( 30, true, true, 301);  
    Salle salle2 = initSalle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



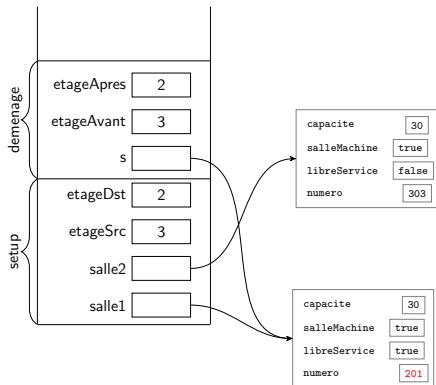
Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = initSalle( 30, true, true, 301);  
    Salle salle2 = initSalle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



Passage de référence en paramètre

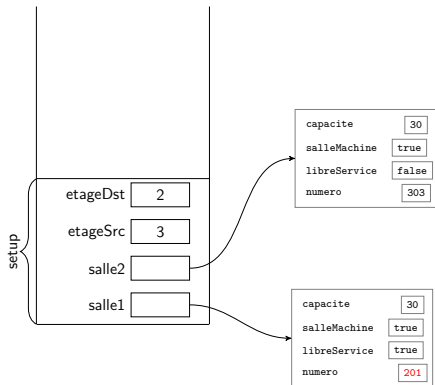
```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = initSalle( 30, true, true, 301);  
    Salle salle2 = initSalle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}
```

```
void setup() {  
    Salle salle1 = initSalle( 30, true, true, 301);  
    Salle salle2 = initSalle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```



Passage de référence en paramètre

```
void demenage(Salle s, int etageAvant, int etageApres) {  
    // si la salle est a l'etage concerne  
    if (s.numero / 100 == etageAvant) {  
        // on demenage ! 301 devient 201  
        s.numero = (s.numero % 100) + (etageApres * 100);  
    }  
}  
  
void setup() {  
    Salle salle1 = initSalle( 30, true, true, 301);  
    Salle salle2 = initSalle( 30, true, false, 303);  
    int etageSrc = 3;  
    int etageDst = 2;  
    println("Numero avant :", salle1.numero);  
    demenage(salle1, etageSrc, etageDst);  
    println("Numero apres :", salle1.numero);  
}
```

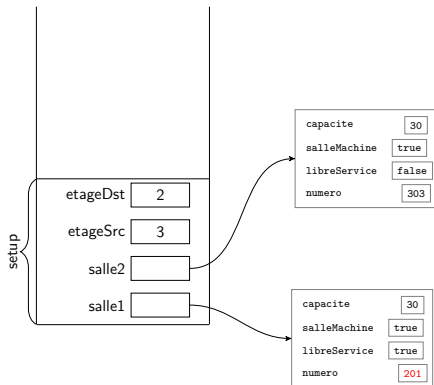
```
Numéro avant : 301  
Numéro après : 201
```



Console

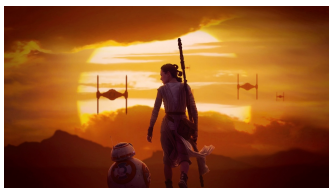


Erreurs



Conclusion

That's all, folks !



Enfin, ce n'est que le début...

Pour l'instant nous n'utiliserons que des objets :

- avec uniquement des **attributs** et des **constructeurs**,
- **que nous définirons**, et pas ceux déjà implémentés dans Java/Processing.

Nous verrons plus tard, et surtout au S2, ce que permet la **programmation objet**.